Задача A. Ancestor. Предок

Имя входного файла:	`ancestor.in`
Имя выходного файла:	`ancestor.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Напишите программу, которая для двух вершин дерева определяет, является ли одна из них предком другой.

Формат входных данных

Первая строка входного файла содержит натуральное число n ($1 \leqslant n \leqslant 100\,000$) — количество вершин в дереве. Во второй строке находится n чисел. При этом i-ое число второй строки определяет непосредственного родителя вершины с номером i. Если номер родителя равен нулю, то вершина является корнем дерева.

В третьей строке находится число m ($1 \leqslant m \leqslant 100\,000$) — количество запросов. Каждая из следующих m строк содержит два различных числа a и b ($1 \leqslant a, b \leqslant n$).

Формат выходных данных

Для каждого из m запросов выведите на отдельной строке число `1`, если вершина a является одним из предков вершины b, и `0` в противном случае.

Примеры

ancestor.in	ancestor.out
6 0 1 1 2 3 3 10 2 1 1 5 4 5 6 1 3 6 5 4 3 1 6 3 6 4 5 1	0 1 0 0 1 0 0 0 0 0

Задача B. Двоичные подъемы

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Задано подвешенное дерево. Найдите для каждой вершины двоичные подъемы: предков, которые находятся от нее на расстоянии 2^k для какого-либо целого k.

Формат входных данных

В первой строке входа задано число n ($1 \leqslant n \leqslant 10^5$) — число вершин дерева. Во второй строке заданы n чисел p_i. Число p_i равно номеру вершины, являющейся предком вершины i (вершины нумеруются с 1) или нулю, если вершина i — корень дерева.

Формат выходных данных

Выведите n строк. В i-й строке выведите номер вершины i и далее после двоеточия список требуемых предков, в порядке увеличения расстояния от i.

Пример

стандартный ввод	стандартный вывод
8 5 8 5 0 4 5 4 1	1: 5 4 2: 8 1 4 3: 5 4 4: 5: 4 6: 5 4 7: 4 8: 1 5

Задача C. Общий предок

Имя входного файла:	`lca.in`
Имя выходного файла:	`lca.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дано подвешенное дерево с корнем в 1-й вершине и M запросов вида “найти у двух вершин наименьшего общего предка”.

Формат входных данных

В первой строке файла записано одно число N — количество вершин. В следующих $N-1$ строках записаны числа. Число x на строке $2 \leqslant i \leqslant n$ означает, что x — отец вершин i. $(x < i)$. На следующей строке число M. Следующие M строк содержат запросы вида (x, y) — найти наименьшего предка вершин x и y Ограничения: $1 \leqslant N \leqslant 5 \cdot 10^4, 0 \leqslant M \leqslant 5 \cdot 10^4$.

Формат выходных данных

M ответов на запросы.

Пример

lca.in	lca.out
5 1 1 2 3 2 2 3 4 5	1 1

Задача D. Расстояние в дереве

Имя входного файла:	`lenpath.in`
Имя выходного файла:	`lenpath.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Вам дано неориентированное дерево (связный граф без циклов) на N вершинах. Требуется отвечать на запросы вида «найти длину кратчайшего пути между данной парой вершин».

Формат входных данных

В первой строке находится одно целое число N ($1 \leqslant N \leqslant 10^5$) — количество вершин в дереве.

В каждой из следующих $N - 1$ строк задано по два числа, разделенных пробелом, u_i, v_i — вершины, соединенные i-ым ребром дерева ($1 \leqslant u_i, v_i \leqslant N$).

В следующей строке задано число M — количество запросов ($0 \leqslant M \leqslant 10^5$).

В каждой из следующих M строк задано по два числа — номера вершин, соответствущих очередному запросу.

Формат выходных данных

Выведите M строк, в каждой из которых содержится одно целое число — ответ на очередной запрос. Ответы необходимо вывести в порядке, соответствующем порядку запросов на входе программы.

Пример

lenpath.in	lenpath.out
3 2 1 3 2 2 2 2 2 1	0 1

Задача E. Самое дешевое ребро

Имя входного файла:	`minonpath.in`
Имя выходного файла:	`minonpath.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Дано подвешенное дерево с корнем в первой вершине. Все ребра имеют веса (стоимости). Вам нужно ответить на M запросов вида найти у двух вершин минимум среди стоимостей ребер пути между ними.

Формат входных данных

В первой строке файла записано одно число — n. (количество вершин).

В следующих $n-1$ строках записаны два числа — x и y. Число x на строке i означает, что x — предок вершины $i+1$, y означает стоимость ребра.

$x \leqslant i, |y| \leqslant 10^6$.

В следующей строке файла записано число m — количество запросов.

Далее m запросов вида (x, y) — найти минимум на пути из x в y ($x \neq y$).

Ограничения: $2 \leqslant n \leqslant 5 \cdot 10^4, 0 \leqslant m \leqslant 5 \cdot 10^4$.

Формат выходных данных

m строк — ответы на запросы.

Пример

minonpath.in	minonpath.out
5 1 2 1 3 2 5 3 2 2 2 3 4 5	2 2

Задача F. Прибавление на пути

Имя входного файла:	`treepathadd.in`
Имя выходного файла:	`treepathadd.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Задано дерево. В каждой вершине есть значение, изначально все значения равны нулю. Требуется обработать запрос прибавления на пути и запрос значения в вершине.

Формат входных данных

В первой строке задано целое число n — число вершин в дереве ($1 \leqslant n \leqslant 3 \cdot 10^5$).

В следующих $n - 1$ строках заданы ребра дерева: по два целых числа v и u в строке — номера вершин, соединенных ребром ($1 \leqslant v, u \leqslant n$).

В следующей строке задано целое число m — число запросов ($1 \leqslant m \leqslant 5 \cdot 10^5$).

Следующие m строк содержат запросы в одном из двух форматов:

- `+ v u d` — прибавить число d во все значения в вершинах на пути от v до u ($1 \leqslant v, u \leqslant n$; $1 \leqslant d \leqslant 10^9$);
- `? v` — вывести значение в вершине v ($1 \leqslant v \leqslant n$).

Формат выходных данных

Выведите ответы на все запросы.

Примеры

treepathadd.in	treepathadd.out
5 1 2 1 3 3 4 3 5 5 + 2 5 1 ? 3 + 1 1 2 ? 1 ? 3	1 3 1

Задача G. Опекуны карнотавров

Имя входного файла:	`carno.in`
Имя выходного файла:	`carno.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Карнотавры очень внимательно относятся к заботе о своем потомстве. У каждого динозавра обязательно есть старший динозавр, который его опекает. В случае, если опекуна съедают (к сожалению, в юрский период такое не было редкостью), забота о его подопечных ложится на плечи того, кто опекал съеденного динамозавра. Карнотавры — смертоносные хищники, поэтому их обычаи строго запрещают им драться между собой. Если у них возникает какой-то конфликт, то, чтобы решить его, они обращаются к кому-то из старших, которому доверяют, а доверяют они только тем, кто является их опекуном или опекуном их опекуна и так далее (назовем таких динозавров суперопекунами). Поэтому для того, чтобы решить спор двух карнотавров, нужно найти такого динозавра, который является суперопекуном для них обоих. Разумеется, беспокоить старших по пустякам не стоит, поэтому спорщики стараются найти самого младшего из динозавров, который удовлетворяет этому условию. Если у динозавра возник конфликт с его суперопекуном, то этот суперопекун сам решит проблему. Если у динозавра нелады с самим собой, он должен разобраться с этим самостоятельно, не беспокоя старших. Помогите динозаврам разрешить их споры.

Формат входных данных

Во входном файле записано число M, обозначающее количество запросов ($1 \leqslant M \leqslant 200000$). Далее на отдельных строках следуют M запросов, обозначающих следующие события:

- `+ v` — родился новый динозавр и опекунство над ним взял динозавр с номером v. Родившемуся динозавру нужно присвоить наименьший натуральный номер, который до этого еще никогда не встречался.
- `- v` — динозавра номер v съели
- `? u v` — у динозавров с номерами u и v возник конфликт и вам надо найти им третейского судью.

Изначально есть один прадинозавр номер 1; гарантируется, что он никогда не будет съеден.

Формат выходных данных

Для каждого запроса типа «?» в выходной файл нужно вывести на отдельной строке одно число — номер самого молодого динозавра, который может выступить в роли третейского судьи.

Пример

carno.in	carno.out
11 + 1 + 1 + 2 ? 2 3 ? 1 3 ? 2 4 + 4 + 4 - 4 ? 5 6 ? 5 5	1 1 2 2 5

Задача H. Генеалогия

Имя входного файла:	`genealogy.in`
Имя выходного файла:	`genealogy.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Во время обсуждений в Парламенте лорды, с похожими взглядами на решение проблемы, обычно объединяются в группы. Как правило, результат обсуждения зависит от решения наиболее влиятельной группы лордов. Именно поэтому подсчёт влиятельности группы является наиболее важной задачей.

Естественно, каждый лорд дорожит древностью своего рода, поэтому влиятельность лорда равна древности его рода. Древность рода лорда — количество предков лорда: его отец, его дед, его прадед, и т.д. Чтобы посчитать влиятельность группы лордов, требуется посчитать количество лордов в группе вместе с их предками. Отметим, что если лорд является предком двух или более лордов в группе, то этот лорд должен быть посчитан только один раз.

Вам дано фамильное дерево лордов (удивительно, но все лорды произошли от одного пра-лорда) и список групп. Для каждой группу найдите её влиятельность.

Формат входных данных

Первая строка входного файла содержит число n — количество лордов ($1 \leqslant n \leqslant 100\,000$). Лорды нумеруются целыми числами от 1 до n. Следующая строка содержит n целых чисел $p_1, p_2, \ldots, p_n$, где p_i — отец лорда с номером i. Если лорд является основателем рода, то p_i равно -1. Гарантируется, что исходные данные формируют дерево. Третья строка входного файла содержит одно число g — количество групп ($1 \leqslant g \leqslant 3\,000\,000$). Следующие g строк содержат описания групп. j-ая строка содержит число k_j — размер j-ой группы, после которого следуют k_j различных чисел — номера лордов, состоящих в j-ой группе. Гарантируется, что сумма всех k_j во входном файле не превосходит $3\,000\,000$.

Формат выходных данных

В выходной файл выведите g строк. В j-ой строке выведите единственное число: влиятельность j-ой группы. Гарантируется, что размер выходного файла не превосходит шести мегабайт.

Примеры

genealogy.in	genealogy.out
4 -1 1 2 3 4 1 4 2 3 4 3 2 3 4 4 1 2 3 4	4 4 4 4
5 2 -1 1 2 3 10 3 3 4 1 3 2 4 3 4 1 3 5 4 1 4 2 2 3 3 1 4 3 1 2 3 3 4 5 1 1 3 1 2 4	4 4 5 2 3 4 1 5 2 3

Задача I. Так и отели

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

N отелей расположены на одной прямой. Координата i-го отеля ($1 \leqslant i \leqslant N$) — x_i.
Так — путешественник, он руководствуется следующими двумя принципами:

1. Он никогда не путешествует на расстояние более чем L за один день.
2. Он никогда не спит под открытым небом. То есть, он должен остановиться в отеле в конце дня.

Вам даны Q запросов. Запрос j ($1 \leqslant j \leqslant Q$) описывается двумя различными целыми числами a_j и b_j. Для каждого запроса нужно найти минимальное количество дней, которые нужно Таку, чтобы доехать от a_j-го отеля до b_j-го отеле, следуя своим принципам. Гарантируется, что он всегда может добраться от a_j-го отеля до b_j-го отеля.

Формат входных данных

Первая строка ввода содержит N — количество гостиниц ($2 \leqslant N \leqslant 10^5$). Вторая строка содержит N чисел — координаты отелей ($1 \leqslant x_i \leqslant 10^9, x_i < x_{i+1}, x_{i+1} - x_i \leqslant L$). Третья строка содержит число L, четвертая строка содержит число Q ($1 \leqslant L \leqslant 10^9, 1 \leqslant Q \leqslant 10^5$). Следующие Q строк содержат описание запросов a_i и b_i ($1 \leqslant a_i, b_i \leqslant N, a_i \neq b_i$).

Формат выходных данных

Выведите Q строк. j-я строка ($1 \leqslant j \leqslant Q$) должна содержать минимальное количество дней, которое нужно Таку, чтобы доехать от a_j-го отеля до b_j-го отеля.

Пример

стандартный ввод	стандартный вывод
9 1 3 6 13 15 18 19 29 31 10 4 1 8 7 3 6 7 8 5	4 2 1 2

Задача J. Декомпозиция

Имя входного файла:	`decomposition.in`
Имя выходного файла:	`decomposition.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Рассмотрим дерево T. Назовем деревом декомпозиции корневое дерево $D(T)$.

Выберем любую из вершин дерева T, назовем ее r. Рассмотрим все компоненты связности дерева T, после удаления вершины r: $S_1, S_2, \ldots, S_k$. Тогда корнем $D(T)$ будет вершина r, а детьми r в $D(T)$ будут $D(S_1), D(S_2), \ldots, D(S_k)$.

Вам задано T. Найдите дерево декомпозиции, высота которого не более 20. Высотой дерева называется максимальное число вершин, которые может содержать простой путь начинающийся в корне.

Формат входных данных

Первая строка содержит n — число вершин дерева T ($1 \leqslant n \leqslant 2 \cdot 10^5$).

Следующие $n-1$ строк содержат ребра дерева. Каждое ребро описывается парой чисел v_i, u_i — концы ребра ($1 \leqslant v_i, u_i \leqslant n$).

Формат выходных данных

Выведите n чисел: i-е число — родитель вершины i в дереве декомпозиции, если вершина является корнем, выведите 0.

Примеры

decomposition.in	decomposition.out
3 1 2 2 3	2 0 2
9 3 2 4 2 1 2 5 1 1 6 7 6 6 8 8 9	0 1 2 2 1 1 6 6 8

Задача K. На далекой Амазонке

Имя входного файла:	`treeeg.in`
Имя выходного файла:	`treeeg.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

В бассейне далёкой реки Амазонки расположены N городов, пронумерованных для удобства целыми числами от 1 до N. Всем известно, что местные леса непроходимы, и передвижение возможно только по рекам. Как следствие, схема соединения городов является деревом.

К несчастью, в этом году в бассейне далёкой Амазонки не на шутку разошлась эпидемия новой болезни — крабового гриппа. То и дело поступает информация о новых заболевших. Поначалу справляться с ней было легко, но вскоре почти все больницы были переполнены, и сейчас пациентов может принимать только госпиталь, находящийся в городе 1.

Для удобства граждан была открыта горячая линия, куда первым делом необходимо обратиться при появлении симптомов крабового (его ещё часто называют раковым) гриппа. Вам необходимо написать программу, которая будет отвечать на обращения пострадавших, учитывая при этом информацию о работающих больницах. Вам ещё повезло, что вы знаете все запросы заранее!

Более формально, поступают запросы трёх видов:

- «`+` v» — госпиталь города v снова может принимать больных. Гарантируется, что в момент перед этим запросом госпиталь города v не работал.
- «`-` v» — госпиталь города v не может больше принимать больных. Гарантируется, что в момент перед этим запросом госпиталь города v работал.
- «`?` v» — заболел человек в городе v, необходимо сообщить ему расстояние до ближайшего города с работающим госпиталем (в идеале неплохо бы ещё и сказать номер этого города, но этим пусть занимаются ваши коллеги). Гарантируется, что в момент такого запроса имеется хотя бы один работающий госпиталь.

Формат входных данных

В первой строке находится единственное число N — количество городов ($1 \leqslant N \leqslant 300\,000$). Следующие $N-1$ строк содержат информацию о соединениях между городами в формате «u v l», что означает соединение между городами u и v длиною l километров ($1 \leqslant u, v \leqslant N$, $1 \leqslant l \leqslant 1000$). Направлением течения можно пренебречь и считать, что время движения зависит только от расстояний.

Далее на отдельной строке записано число Q — количество запросов. Следующие Q строк содержат описание запросов в формате «c v», где c — это один из трёх символов «`+`», «`-`» и «`?`», а v — номер города ($1 \leqslant v \leqslant N$).

Формат выходных данных

Для каждого запроса вида «`?` v» выведите на отдельной строке одно число — расстояние в километрах до ближайшего города с работающим госпиталем.

Примеры

treeeg.in	treeeg.out
5 1 2 2 2 3 3 3 4 1 3 5 4 5 ? 4 + 5 ? 3 - 1 ? 2	6 4 7

Задача L. БДБД

Имя входного файла:	`lwdb.in`
Имя выходного файла:	`lwdb.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Большая Древесная База Данных создана для того, чтобы в ней можно было надежно сохранить и раскрасить любое дерево. В новой версии БДБД запланирован новый функционал, для реализации которого потребуется вновь переосмыслить теорию графов.

В БДБД хранится взвешенное дерево. В языке запросов Системы Управления Большой Древесной Базы Данных (СУБДБД) предусмотрены два вида запросов:

1. «1 v d c» — покрасить все вершины, находящиеся на расстоянии не более d от вершины v, в цвет c. Все вершины изначально окрашены в цвет с номером 0.
2. «2 v» — вывести цвет вершины v.

Необходимо запрограммировать работу СУБДБД и ответить на все запросы пользователя.

Формат входных данных

В первой строке число N ($1 \leqslant N \leqslant 10^5$) — количество вершин дерева.

Следующие $N-1$ строк содержат описание ребер, по три числа в строке a_i, b_i, w_i ($1 \leqslant a_i, b_i \leqslant N$, $a_i \neq b_i$, $1 \leqslant w_i \leqslant 10^4$), где i-е ребро имеет вес w_i и соединяет вершины a_i и b_i.

В следующей строке число Q ($1 \leqslant Q \leqslant 10^5$) — число запросов. В каждой из Q следующих строк запросы одного из двух видов:

1. Числа 1, v, d, c ($1 \leqslant v \leqslant N$, $0 \leqslant d \leqslant 10^9$, $0 \leqslant c \leqslant 10^9$).
2. Числа 2, v ($1 \leqslant v \leqslant N$).

Все числа во входных данных целые.

Формат выходных данных

Для каждого запроса второго типа необходимо вывести в отдельной строке цвет запрошенной вершины.

Примеры

lwdb.in	lwdb.out
5 1 2 30 1 3 50 3 4 70 3 5 60 8 1 3 72 6 2 5 1 4 60 5 2 3 2 2 1 2 144 7 2 4 2 5	6 6 0 5 7

Задача M. Гоша и праздники

Имя входного файла:	`events.in`
Имя выходного файла:	`events.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мегабайт

Как известно, жители планеты Иннополис — очень педантичные люди. И даже когда дело касается праздников, они всегда хотят быть уверенными в том, что все пройдёт как по маслу. Так, расписание празднований всех событий на этой планете составлено почти на три миллиона лет вперёд! Гоша — большой любитель праздников. Он решил прилететь в какой-то из городов планеты Иннополис и посетить как можно больше праздников.

На планете Иннополис n городов, соединённых $n - 1$ двунаправленными дорогами так, что из любого города планеты можно добраться до любого другого, возможно, посещая другие города. Каждое событие на Иннополисе характеризуется номером города c_i, в котором оно будет отпраздновано, и номером дня d_i, в который его будут праздновать.

Гоша настолько везучий человек, что день его прибытия на планету имеет номер 0 в календаре планеты Иннополис, причём исходно он может прилететь в любой город планеты. Гоша решил узнать, какое максимальное количество праздников он может посетить на этой планете. Для этого он обратился за помощью к вам.

Формат входных данных

В первой строке входного файла задано одно число n ($n \geqslant 1$) — количество городов Иннополиса.

В следующих $n - 1$ строках заданы описания дорог, каждая дорога задается числами a_i, b_i и l_i ($1 \leqslant a_i, b_i, \leqslant n$; $l_i \geqslant 1$) — номера городов, которые соединяет дорога и число дней, необходимых на ее преодоление.

В следующей строке задано число m ($m \geqslant 1$) — число праздников на планете.

В следующих m строках заданы пары чисел c_i и d_i ($1 \leqslant c_i \leqslant n$; $d_i \geqslant 1$) — номер города и номер дня, в который пройдёт i-й праздник.

Формат выходных данных

В единственной строке выходного файла выведите одно число — максимальное количество праздников, которое может посетить Гоша.

Система оценки

Номер подзадачи	Баллы	Ограничения				Комментарии
		n	m	l_i	d_i	
0	0					Примеры из условия.
1	14	$n \leqslant 100$	$m \leqslant 9$	$l_i \leqslant 100$	$d_i \leqslant 100$	Баллы начисляются, если все тесты пройдены.
2	17	$n \leqslant 2000$	$m \leqslant 2000$	$l_i \leqslant 5000$	$d_i \leqslant 5000$	Баллы начисляются, если все тесты этой и предыдущих подзадач пройдены.
3	28	$n \leqslant 5000$	$m \leqslant 5000$	$l_i \leqslant 10^9$	$d_i \leqslant 10^9$	Баллы начисляются, если все тесты этой и предыдущих подзадач пройдены.
4	22	$n \leqslant 10^5$	$m \leqslant 10^5$	$l_i \leqslant 10^9$	$d_i \leqslant 10^9$	Баллы начисляются, если все тесты этой и предыдущих подзадач пройдены.
5	19	$n \leqslant 2 \cdot 10^5$	$m \leqslant 2 \cdot 10^5$	$l_i \leqslant 10^9$	$d_i \leqslant 10^9$	Баллы на каждый тест начисляются отдельно, если все тесты предыдущих подзадач пройдены.

Примеры

events.in	events.out
4 1 2 1 2 3 1 2 4 3 4 1 3 2 4 3 1 4 5	3

Задача N. Близкие вершины

Имя входного файла:	`close-vertices.in`
Имя выходного файла:	`close-vertices.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Дано дерево из n вершин. Каждое ребро имеет неотрицательный вес. Длиной пути между двумя вершинами называется количество ребер в пути. Весом пути называется суммарный вес всех входящих в него ребер.

Две вершины называются близкими, если существует путь между двумя этими вершинами длины не более l и также существует путь между ними веса не более w. Определите количество близких пар вершин.

Формат входных данных

В первой строке записаны три целых числа n, l и w ($1 \leqslant n \leqslant 10^5, 1 \leqslant l \leqslant n, 0 \leqslant w \leqslant 10^9$). Далее в $n-1$ строках дано описание ребер дерева. В i-той строке записано два целых числа p_i, w_i ($1 \leqslant p_i < (i+1), 0 \leqslant w_i \leqslant 10^4$), которые обозначают, что i-ое ребро соединяет вершину $(i+1)$ и p_i и имеет вес w_i.

Считайте, что вершины дерева пронумерованы от 1 до n некоторым образом.

Формат выходных данных

Выведите единственное целое число — количество близких пар.

Примеры

close-vertices.in	close-vertices.out
4 4 6 1 3 1 4 1 3	4
6 2 17 1 3 2 5 2 13 1 6 5 9	9